

日本国特許庁  
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されて  
いる事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed  
with this Office

出願年月日

Date of Application: 2002年 8月19日

出願番号

Application Number: 特願2002-238399

[ ST.10/C ]:

[ J P 2002-238399 ]

出願人

Applicant(s): 富士通株式会社

2003年 1月 7日

特許庁長官  
Commissioner,  
Japan Patent Office

太田信一郎



出証番号 出証特2002-3103565

Attorney Docket No. 1448.1040

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re Patent Application of:

Hideo MIYAKE, et al.

Application No.:

Group Art Unit:

Filed: June 25, 2003

Examiner:

For: METHOD OF AND APPARATUS FOR CREATING LOAD MODULE AND COMPUTER PRODUCT

**SUBMISSION OF CERTIFIED COPY OF PRIOR FOREIGN  
APPLICATION IN ACCORDANCE  
WITH THE REQUIREMENTS OF 37 C.F.R. § 1.55**

Commissioner for Patents  
PO Box 1450  
Alexandria, VA 22313-1450

Sir:

In accordance with the provisions of 37 C.F.R. § 1.55, the applicant(s) submit(s) herewith a certified copy of the following foreign application:

Japanese Patent Application No(s). 2002-187230 and 2002-238399

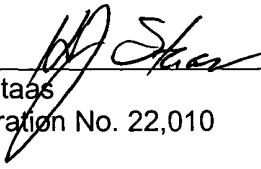
Filed: June 27, 2002 and August 19, 2002

It is respectfully requested that the applicant(s) be given the benefit of the foreign filing date(s) as evidenced by the certified papers attached hereto, in accordance with the requirements of 35 U.S.C. § 119.

Respectfully submitted,

STAAS & HALSEY LLP

Date: June 25, 2003

By: 

H. J. Staas  
Registration No. 22,010

1201 New York Ave, N.W., Suite 700  
Washington, D.C. 20005  
Telephone: (202) 434-1500  
Facsimile: (202) 434-1501

【書類名】 特許願

【整理番号】 0240872

【提出日】 平成14年 8月19日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 15/16

【発明の名称】 ロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置

【請求項の数】 9

【発明者】

【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通  
株式会社内

【氏名】 三宅 英雄

【発明者】

【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通  
株式会社内

【氏名】 上方 輝彦

【発明者】

【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通  
株式会社内

【氏名】 畑上 謙吾

【特許出願人】

【識別番号】 000005223

【氏名又は名称】 富士通株式会社

【代理人】

【識別番号】 100104190

【弁理士】

【氏名又は名称】 酒井 昭徳

【手数料の表示】

【予納台帳番号】 041759

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9906241

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 ロードモジュール生成方法、ロードモジュール生成プログラム  
およびロードモジュール生成装置

【特許請求の範囲】

【請求項1】 複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成方法において、

第1のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第1のメモリ空間構築工程と、

前記第1のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第1のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第1のメモリ空間内アドレス解決工程と、

第2のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第2のメモリ空間構築工程と、

前記第2のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第2のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第2のメモリ空間内アドレス解決工程と、

前記第1のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第2のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決工程と、

を含んだことを特徴とするロードモジュール生成方法。

【請求項2】 前記メモリ空間間アドレス解決工程では、前記第1のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第2のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスの、前記第2のメモリ空間構築工程で構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする請求項1に記載のロードモジュール生成方法。

【請求項3】 前記第1のメモリ空間構築工程では、前記第1のプロセッサからのみ参照可能なメモリ領域と、前記第1のプロセッサおよび前記第2のプロ

セッサから参照可能なメモリ領域とを構築することを特徴とする請求項1または請求項2に記載のロードモジュール生成方法。

【請求項4】 複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成プログラムにおいて、

第1のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第1のメモリ空間構築工程と、

前記第1のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第1のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第1のメモリ空間内アドレス解決工程と、

第2のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第2のメモリ空間構築工程と、

前記第2のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第2のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第2のメモリ空間内アドレス解決工程と、

前記第1のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第2のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決工程と、

をコンピュータに実行させることを特徴とするロードモジュール生成プログラム。

【請求項5】 前記メモリ空間間アドレス解決工程では、前記第1のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第2のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスの、前記第2のメモリ空間構築工程で構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする請求項4に記載のロードモジュール生成プログラム。

【請求項6】 前記第1のメモリ空間構築工程では、前記第1のプロセッサからのみ参照可能なメモリ領域と、前記第1のプロセッサおよび前記第2のプロセッサから参照可能なメモリ領域とを構築することを特徴とする請求項4または

請求項5に記載のロードモジュール生成プログラム。

【請求項7】 複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成装置において、

第1のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第1のメモリ空間構築手段と、

前記第1のメモリ空間構築手段により構築された各メモリ領域のアドレスにもとづいて、前記第1のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第1のメモリ空間内アドレス解決手段と、

第2のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第2のメモリ空間構築手段と、

前記第2のメモリ空間構築手段により構築された各メモリ領域のアドレスにもとづいて、前記第2のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第2のメモリ空間内アドレス解決手段と、

前記第1のメモリ空間内アドレス解決手段によりアドレスが解決されなかったシンボルのアドレスを、前記第2のメモリ空間内アドレス解決手段により算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決手段と、

を備えたことを特徴とするロードモジュール生成装置。

【請求項8】 前記メモリ空間間アドレス解決手段は、前記第1のメモリ空間内アドレス解決手段によりアドレスが解決されなかったシンボルのアドレスを、前記第2のメモリ空間内アドレス解決手段により算出された当該シンボルのアドレスの、前記第2のメモリ空間構築手段により構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする請求項7に記載のロードモジュール生成装置。

【請求項9】 前記第1のメモリ空間構築手段は、前記第1のプロセッサからのみ参照可能なメモリ領域と、前記第1のプロセッサおよび前記第2のプロセッサから参照可能なメモリ領域とを構築することを特徴とする請求項7または請求項8に記載のロードモジュール生成装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

この発明は、複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置に関する。

【0002】

【従来の技術】

近年の計算機システムでは、複数のプロセッサを搭載することでシステムの処理能力を向上させるべく、「分散メモリ型マルチプロセッサ方式（Distributed-Memory Multiprocessors）」が採用されることがある。

【0003】

図1は、分散メモリ型マルチプロセッサ方式にもとづいた計算機システムを模式的に示す説明図である。図示するように、プロセッサ（PROCESSOR）101とメモリ（MEMORY）102とから構成されるPE（Processing Element：プロセッサ要素）100がn個、相互接続網（INTERCONNECTION NETWORK）103により接続されている。

【0004】

また、図2は上記システムにおけるメモリ空間の定義例を模式的に示す説明図である。図示するように個々のプロセッサ101は、同じPE100内のメモリ102だけを読み書きすることができる。

【0005】

そしてこのようなシステムにおいては、MPI（Message-Passing Interface）などのプロセッサ間通信機構を用いることで、SPMD（Single-Program, Multiple-Data）プログラミングにもとづくプログラムが実行されることが多い。

【0006】

図3は上記プログラムの一例を示す説明図である。図示するプログラムはn個

のメモリ102にそれぞれ格納され、n個のプロセッサ101によりそれぞれ実行される。プログラムは同一でも、PE100のID（番号）により処理が分岐するので、n個のPE100による並列処理が実現される。

#### 【0007】

たとえば図示するプログラムでは、「my\_rank」が上記IDを示す変数であり、 $my\_rank = 0$ 以外のPEではif以下の処理が、 $my\_rank = 0$ のPEではelse以下の処理が、それぞれ実行されることになる。

#### 【0008】

また、図4は上記プログラムのロードモジュールの生成手順を示すフローチャートである。まずコンパイラにより、プログラムのソース記述をアセンブリ記述に変換し（ステップS401～S403）、次にアセンブラにより、上記アセンブリ記述からオブジェクトを生成する（ステップS404～S406）。そして、上記により生成された複数のオブジェクトをリンクにより結合することで（ステップS407～S410）、上記プログラムのロードモジュールを生成する。

#### 【0009】

ところで、図1のような分散メモリ型マルチプロセッサ方式にもとづくシステムは、従来は半導体集積技術の限界から、複数のチップ（および複数のボード）により構成されてきた。しかしながら、近年の半導体集積技術の向上により、複数のPEを一つのチップに収めることができることが可能となっている。

#### 【0010】

この場合、相互接続網を介したPE間のデータの受け渡しはパケット伝送方式ではなく、共有メモリにデータを直接ストア／共有メモリからデータを直接ロードすることで、より高速におこなうことができる。このように、複数のプロセッサから読み書きされる共有メモリを設ける方式を、「分散共有メモリ型マルチプロセッサ方式」と呼ぶ。

#### 【0011】

図5は、分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムを模式的に示す説明図である。図1に示した分散メモリ型マルチプロセッサ方式との差異は、メモリ502に、他のPE内のプロセッサからも読み書きできるS

M (Shared Memory: 共有メモリ) と、同一のPE内のプロセッサからしか読み書きできないLM (Local Memory: 固有メモリ)との2種類ある点である。

#### 【0012】

また、図6は上記システムにおけるメモリ空間の定義例を示す説明図である。図中、たとえば1番のPE (PE # 1) のSMは、0番のPE (PE # 0) のメモリ空間および1番のPE (PE # 1) のメモリ空間に重複して割り当てられている。

#### 【0013】

仮に、PE # 1のSMがPE # 0のメモリ空間では $0 \times 3000$ 以下、PE # 1のメモリ空間では $0 \times 2000$ 以下のアドレスに割り当てられていたとする、たとえばPE # 0が $0 \times 2000$ にデータを書き込み、PE # 1が $0 \times 3000$ からデータを読み出すことで、PE # 0とPE # 1との間で上記データを授受できたことになる。

#### 【0014】

なお、図示する例ではPE # 0のみが、他のすべてのPEのSMを参照・変更することができる。一方PE # 1～# nの各メモリ空間には、物理的に他のPEに属するメモリが割り当てられていないので、これらのPEは同一PE内のLMおよびSMを参照・変更するのみである。

#### 【0015】

そして、このような分散共有メモリ型マルチプロセッサ方式の計算機システムでも、分散メモリ型マルチプロセッサ方式と同様、図3に示したようなSPMDプログラミングにもとづくプログラムを実行することは可能である。

#### 【0016】

##### 【発明が解決しようとする課題】

しかしながら、分散メモリ型マルチプロセッサ方式にせよ、分散共有メモリ型マルチプロセッサ方式にせよ、個々のプロセッサが実行するのはプログラムの一部（以下では「部分プログラム」という）であるにもかかわらず、各PEにはプログラムの全体が配分されるので、それだけの容量のメモリを用意しなければな

らず、コストがかさんでしまうという問題があった。

#### 【0017】

この問題は、少なくとも分散メモリ型マルチプロセッサ方式のシステムについては、プログラムをSPMDでなくMPMD (Multiple-Program, Multiple-Data) プログラミングにもとづいて作成することで回避可能である。

#### 【0018】

MPMDにもとづくプログラミングでは、SPMDのように各PEにより実行される部分プログラムをすべて結合したようなプログラムでなく、端的にそれぞれのPE向けのプログラムを作成する。図7はPE#0、図8はPE#1～#n向けのプログラムの一例をそれぞれ示す説明図である。図示するように各PE用のプログラムには、他PE用の部分プログラムが含まれないので、その分メモリの容量を小さくすることができる。なお、これらのプログラムのロードモジュールは、図4に示した手順により生成される。

#### 【0019】

一方、分散共有メモリ型マルチプロセッサ方式では、上述のようにある場所に格納された同一のデータを複数のPEが参照・変更するが、メモリ空間上のそのアドレスはPEごとに異なっている。したがって、各PE向けのプログラムについてリンクでアドレスを解決する際には、対象物が同一でもPEごとに異なるアドレスに変換しなければならないが、従来技術のリンクにはこうした機能がなかった。

#### 【0020】

そのため分散共有メモリ型マルチプロセッサ方式の計算機システムで動作するプログラムは、SPMDプログラミングによってしか作成することができず、実行時にはスキップされる部分プログラムが各PEに多数配分され、無駄な部分プログラムを保持するために多くのメモリが必要になってしまうという問題があつた。

#### 【0021】

この発明は上記従来技術による問題を解決するため、分散共有メモリ型マルチ

プロセッサ方式を採用する計算機システムにおいても、より少ないメモリ容量で動作するプログラムのロードモジュールを生成することが可能なロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置を提供することを目的とする。

## 【0022】

## 【課題を解決するための手段】

上述した課題を解決し、目的を達成するため、この発明にかかるロードモジュール生成方法、ロードモジュール生成プログラムまたはロードモジュール生成装置は、複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成方法において、第1のプロセッサにより実行されるプログラムおよび第2のプロセッサにより実行されるプログラムについて、それぞれオブジェクトを結合して所定のメモリ領域を構築のうえ各シンボルのアドレスを算出するとともに、第1のプロセッサにより実行されるプログラム中アドレスが解決されなかったシンボルのアドレスを、第2のプロセッサにより実行されるプログラム中での当該シンボルのアドレスにもとづいて算出することを特徴とする。

## 【0023】

また、この発明にかかるロードモジュール生成方法、ロードモジュール生成プログラムまたはロードモジュール生成装置は、第1のプロセッサにより実行されるプログラム中アドレスが解決されなかったシンボルのアドレスを、第2のプロセッサにより実行されるプログラム中での当該シンボルのアドレスの、当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする。

## 【0024】

また、この発明にかかるロードモジュール生成方法、ロードモジュール生成プログラムまたはロードモジュール生成装置は、第1のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する際、第1のプロセッサからのみ参照可能なメモリ領域と、第1のプロセッサおよび第2のプロセッサから参照可能なメモリ領域とを構築することを特徴とする。

## 【0025】

これらの発明によって、MPMDプログラミングにもとづいて作成された各プロセッサ用のソースプログラムから、分散共有メモリ型マルチプロセッサ方式を採用する計算機でも実行可能なそのロードモジュールを生成することができる。逆にいえば、分散共有メモリ型マルチプロセッサ方式の計算機についても、MPMDプログラミングによってプログラムを作成することができる。

## 【0026】

## 【発明の実施の形態】

以下に添付図面を参照して、この発明にかかるロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置の好適な実施の形態を詳細に説明するが、その前に本発明の基本方針を簡単に説明する。

## 【0027】

## (基本方針)

図9および図10は、分散共有メモリ型マルチプロセッサ方式の計算機システムを前提とする、MPMDプログラミングにもとづくプログラムの一例を示す説明図である。図9はPE#0用、図10はPE#1用であり、PE#0からPE#1に必要なデータを渡して、所定の処理を依頼した後、その結果を受け取るためのプログラムである。

## 【0028】

すなわちまずPE#0において、変数`in put`を読み出して、その値を変数`in`に書き込み（図9 Th0-1）、次にPE#1の関数`Th1`の実行を指示する（図9 Th0-2）。これを受けたPE#1では、`Th1`の中で変数`in`を入力として関数`f1`を呼び出し、その実行結果を変数`out`に書き込む（図10 Th1-1）。その後、PE#0は変数`out`を読み出し、その値を変数`out put`に書き込む（図9 Th0-3）。

## 【0029】

なお、実際のプログラムではPE#1に処理を依頼した後（すなわちTh0-2の後）、PE#0はPE#1とは無関係な別の処理に移行するが、ここでは簡略化してPE#0-PE#1間の連携部分のみを示している。

## 【0030】

上述のように、従来技術の言語処理系ではこれらのソースプログラムについて、実際に実行可能なロードモジュールを生成することができない。たとえば図9に示したP E # 0用のプログラムで、`extern`宣言されている変数`in`および`out`は、図10に示したP E # 1用のプログラムで定義されているため、P E # 0用のプログラムをリンクした時点ではアドレスは不定である。

## 【0031】

P E # 1用のプログラムをリンクすると上記変数のアドレスは確定するが、判明するのはあくまでP E # 1のメモリ空間上でのアドレスであり、当該アドレスで指し示される物理的な記憶領域の、P E # 0のメモリ空間上におけるアドレスは依然不明である。

## 【0032】

そこで、以下に説明する実施の形態のような計算式を用いて、P E # 1のメモリ空間上でのアドレスから、P E # 0のメモリ空間上でのアドレスを割り出してやることで、P E # 0用のプログラムで未解決シンボルとして残った、変数`in`および`out`のアドレスを解決する。

## 【0033】

図11は、図9および図10に示したプログラムが本発明によりアドレス解決された後の状況を模式的に示す説明図である。同一の変数`in`/`out`が、P E # 0用のプログラムでは`0x3000/0x3004`に、P E # 1用のプログラムでは`0x2000/0x2004`に、それぞれ置換されていることが分かる。

## 【0034】

なお、図中「`text area`」とはプログラムの命令列を保持する領域であり、「`data area`」とはプログラムから読み書きされるデータのうち、非共有データすなわち当該プログラムを実行するP E以外の他のP Eからは参照・変更されることのないデータを保持する領域である。これらの領域は物理的には各P EのLMに配置され、他のP Eからの参照や変更はできない。

## 【0035】

また、図中「`shared data area #k`」( $0 \leq k \leq n$ )は、

いずれも共有データを保持する領域である。これらの領域は物理的には、それぞれk番目のPEのSMに配置され、他のPEから参照・変更される可能性があるものとする。

## 【0036】

たとえば変数inの格納場所は、実際にはPE#1のSM上的一点であり、同じ場所にPE#0は0x3000、PE#1は0x2000のアドレスを割り当てているので、いずれのPEからもその値を参照・変更することができる。これにより、共有メモリを介したPE間でのデータの授受が可能となっている。

## 【0037】

(実施の形態)

次に図12は、本発明の実施の形態にかかるロードモジュール生成装置のハードウェア構成の一例を示すブロック図である。

## 【0038】

図中、まずCPU1201は装置全体の制御を司る。ROM1202はブートプログラムなどを記憶している。RAM1203はCPU1201のワークエリアとして使用される。HDD1204は、CPU1201の制御にしたがってHD1205に対するデータのリード／ライトを制御する。HD1205は、HDD1204の制御にしたがって書き込まれたデータを記憶する。

## 【0039】

FDD1206は、CPU1201の制御にしたがってFD1207に対するデータのリード／ライトを制御する。FD1207は、FDD1206の制御にしたがって書き込まれたデータを記憶したり、記憶しているデータをFDD1206の磁気ヘッドに読み取らせたりする。着脱可能な記録媒体としては、FD1207のほかCD-ROM、CD-R、CD-RW、MO、DVD(Digital Versatile Disk)、メモリカードなどが考えられる。

## 【0040】

ディスプレイ1208は、たとえばCRT、TFT液晶ディスプレイ、プラズマディスプレイなどであって、カーソルやウィンドウをはじめ、文書、画像などの各種データを表示する。ネットワークI/F1209は、イーサネット(R)

ケーブル1210を通じてLANに接続されるとともに、LANと装置内部とのデータの送受信を司る。

#### 【0041】

キーボード1211は、文字、数値、各種指示などの入力のためのキーを備え、装置内部へのデータの入力を司る。タッチパネル式の入力パッドやテンキーなどであってもよい。マウス1212は、カーソルの移動や範囲選択などを司る。ポインティングデバイスとして同様の機能を備えるものであれば、トラックボール、ジョイスティック、十字キー、ジョグダイヤルなどであってもよい。なお、上記各部はバスまたはケーブル1200により接続されている。

#### 【0042】

次に、図13は本発明の実施の形態にかかるロードモジュール生成装置の構成を機能的に示すブロック図である。同図に示す各機能部は、具体的には図12に示したHD1205、FD1207などに格納されたプログラム、具体的にはコンパイラ、アセンブリおよびリンクの三つのプログラムを、CPU1201がRAM1203に読み出して実行することにより実現される。

#### 【0043】

図中、1300～1302はコンパイラにより実現され、プログラムのソース記述をアセンブリ記述に変換する機能部である。その詳細な機能は従来技術によるコンパイラと同一である。

#### 【0044】

すなわち第1解析部1300は、図4に示したステップS401の処理をおこなう機能部であり、指定されたプログラムのソース記述を読み込んで字句解析および構文解析をおこなうとともに、当該プログラムをコンパイラの内部表現へと変換する。

#### 【0045】

次に、命令列生成部1301は図4に示したステップS402の処理をおこなう機能部であり、上記内部表現にもとづいてプログラムの動作を実現する命令列を生成するとともに、当該命令列をコンパイラの内部情報に付加する。

#### 【0046】

次に、アセンブリ記述出力部1302は図4に示したステップS403の処理をおこなう機能部であり、コンパイラの内部表現および付加されている命令列にもとづいて、上記プログラムのアセンブリ記述を出力する。

#### 【0047】

また、図13中1303～1305はアセンブラにより実現され、コンパイラから出力されたアセンブリ記述をさらにオブジェクトに変換する機能部である。その詳細な機能は従来技術によるアセンブラと同一である。

#### 【0048】

すなわち第2解析部1303は、図4に示したステップS404の処理をおこなう機能部であり、コンパイラのアセンブリ記述出力部1302から出力されたアセンブリ記述を読み込んで、字句解析をおこなうとともにアセンブラの内部表現へと変換する。

#### 【0049】

次に、バイナリ・コード生成部1304は図4に示したステップS405の処理をおこなう機能部であり、アセンブラの内部表現にもとづいてバイナリ・コード（命令コードを含む）を生成し、当該コードをアセンブラの内部情報に付加する。

#### 【0050】

次に、オブジェクト出力部1305は図4に示したステップS406の処理をおこなう機能部であり、アセンブラの内部表現および付加されているバイナリ・コードにもとづいて、上記プログラムのオブジェクトを出力する。

#### 【0051】

また、図13中1306～1311はリンクにより実現され、アセンブラから出力されたオブジェクトを結合して実行可能なロードモジュールを出力する機能部である。1307以下の各部の機能については、後述するフローチャートで説明するが、1306についてのみ先に説明する。

#### 【0052】

図14は、メモリ空間定義情報記憶部1306に保持されるメモリ空間定義情報の内容を模式的に示す説明図である。メモリ空間定義情報とは、上述の「t e

`text area`」「`data area`」などの各メモリ領域を、メモリ空間内のどのアドレスに配置するかをPEごとに定義したものである。

#### 【0053】

図中、たとえばPE#0のメモリ空間上で $0x0000$ から $0xf f f$ までのアドレスが指示する場所は、物理的にはPE#0のLM上に存在し、ここにPE#0用のプログラムの「`text area`」が配置される。同様に、PE#1のメモリ空間上で $0x0000$ から $0xf f f$ までのアドレスが指示する場所は、物理的にはPE#1のLM上に存在し、ここにPE#1用のプログラムの「`text area`」が配置される。

#### 【0054】

また、PE#0のメモリ空間上で $0x3000$ から $0x3ff$ までのアドレスが指示する場所と、PE#1のメモリ空間上で $0x2000$ から $0x2ff$ までのアドレスが指示する場所とは同一であり、物理的にはPE#1のSM上に存在する。そして、ここに「`shared data area #1`」、すなわちPE#0からも参照・変更可能なPE#1の共有データが配置される。

#### 【0055】

次に、図15は本発明の実施の形態にかかるロードモジュール生成装置における、図9および図10に示したプログラムのロードモジュール生成処理の手順を示すフローチャートである。もっとも、コンパイラおよびアセンブラーによる処理は従来技術と同様であるので、同図にはリンカによる処理の手順のみを示している。

#### 【0056】

まず、リンカにより実現されるオブジェクト読み込み部1307が、アセンブラーのオブジェクト出力部1305から出力されたオブジェクトのうち、k番目( $0 \leq k \leq n$ )のPE用のものをリンカの内部表現として読み込む(ステップS1501)。

#### 【0057】

次にメモリ空間構築部1308が、リンカの内部表現において、k番目のPEの各メモリ領域(上述の「`text area`」や「`data area`」など

) を形成し、リンクの内部表現として付加する(ステップS1502)。

#### 【0058】

次にメモリ空間内アドレス解決部1309が、リンクの内部表現において、k番目のPEのメモリ空間内の各メモリ領域のアドレス解決をおこなう(ステップS1503)。そして、ここまで処理をPE#0(k=0)からPE#n(k=n)のすべてについておこなう。

#### 【0059】

次にメモリ空間間アドレス解決部1310は、上記の処理で得られた各PEのメモリ空間イメージと、図14に示したメモリ空間定義情報とを参照して、ステップS1503による各メモリ空間内でのアドレス解決では未解決のまま残ったシンボルについて、メモリ空間をまたがったアドレス解決をおこなう(ステップS1504)。

#### 【0060】

メモリ空間をまたがったアドレス解決とは、具体的にはたとえば図9に示したPE#0用のプログラムにおいて、PE#1用のプログラムで宣言されているためにアドレスが解決できない変数inおよびoutについて、当該変数のPE#1のメモリ空間上でのアドレスから、PE#0のメモリ空間上でのアドレスを算出する処理である。

#### 【0061】

あるシンボルのあるPEにおけるアドレスを、他PEにおけるアドレスから算出するための計算式は下記の通りである。

```
symbol address = self base address + offset
offset
```

```
ただし offset = other's PE symbol address
- other's PE base address
```

#### 【0062】

たとえば変数outのPE#0におけるアドレスは、offset=4 (=0x2004-0x2000)であることから、symbol address=0x3004 (=0x3000+0x0004)となる。すなわち図14のメモ

リ空間定義情報から、少なくともPE#0のメモリ空間上でPE#1との共有データが配置される開始アドレスは分かるので、変数 $o\ u\ t$ のオフセットを当該開始アドレスに足し合わせることで、PE#0にとっての当該変数のアドレスを割り出している。

#### 【0063】

メモリ空間内アドレス解決部1309およびメモリ空間間アドレス解決部1310によるアドレス解決の後は、未解決シンボルは存在しないはずなので、次にロードモジュール出力部1311がリンクの内部表現にもとづいて、k番目のPE用のプログラムのロードモジュールを出力する（ステップS1505）。そして、PE#0（k=0）からPE#n（k=n）のすべてについて同様にロードモジュールを出力した時点で、全PEについてソースからロードモジュールまでのプログラムの変換処理が終了する。

#### 【0064】

なお、図6に示したメモリ空間定義例のもとでは、物理的に他PEのメモリに存在する変数を参照・変更するプログラムはPE#0用のものに限定される（PE#1～#nの各メモリ空間には他PEのメモリが割り当てられていないため、これらのPEについて、他PEの変数を参照・変更するようなプログラムは作成することができない）。したがって、PE#1～#n用のプログラムのリンク時には、ステップS1503の後未解決シンボルは残らないはずなので、ステップS1504によるアドレス解決は不要である。

#### 【0065】

もっとも、図6のようにPE#0のみが他のPEのメモリを読み書きできるようにする必然性はなく（図6のような定義は一例に過ぎない）、PE#0以外のPEからも他PEのメモリを参照・変更できるようにしてもよい。

#### 【0066】

その場合PE#0以外でも、ステップS1503の処理だけでは解決できないシンボルが発生しうるため、たとえば図16に示すように、PE#1～#nについてもステップS1504によるアドレス解決をおこなう。PE#0について行ったのと同様の処理を他PEについてもおこなうというのみで、図15の手順と

処理の内容が異なるわけではない。逆に図15の手順は、図16においてPE#0以外のPEにつきステップS1504を省略できる特別の場合であるとみることもできる。

#### 【0067】

以上説明したように、本発明によれば分散共有メモリ型マルチプロセッサ方式に対応した、MPMDプログラミングにもとづくプログラムのロードモジュールを生成することが可能である。逆にいえば本発明により、分散共有メモリ型マルチプロセッサ方式を前提とするプログラムを、MPMDプログラミングにより作成することができるので、各PEにはそこで実行される部分プログラムのみを配置することができ、チップに搭載するメモリの容量を少なくすることができる。

#### 【0068】

なお、本実施の形態におけるロードモジュール生成方法は、あらかじめ用意されたプログラム（コンパイラ、アセンブラーおよびリンク）がパーソナルコンピュータ、ワークステーションなどの各種のコンピュータ上で実行されることにより実現されるが、このプログラムはHD、FD、CD-ROM、MO、DVDなどのコンピュータで読み取り可能な各種の記録媒体に記録され、当該記録媒体によって配布することができるほか、インターネットなどのネットワークを介して配布することも可能である。

#### 【0069】

（付記1）複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成方法において、

第1のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第1のメモリ空間構築工程と、

前記第1のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第1のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第1のメモリ空間内アドレス解決工程と、

第2のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第2のメモリ空間構築工程と、

前記第2のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづ

いて、前記第2のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第2のメモリ空間内アドレス解決工程と、

前記第1のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第2のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決工程と、  
を含んだことを特徴とするロードモジュール生成方法。

#### 【0070】

(付記2) 前記メモリ空間間アドレス解決工程では、前記第1のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第2のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスの、前記第2のメモリ空間構築工程で構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする付記1に記載のロードモジュール生成方法。

#### 【0071】

(付記3) 前記第1のメモリ空間構築工程では、前記第1のプロセッサからのみ参照可能なメモリ領域と、前記第1のプロセッサおよび前記第2のプロセッサから参照可能なメモリ領域とを構築することを特徴とする付記1または付記2に記載のロードモジュール生成方法。

#### 【0072】

(付記4) 前記第1のプロセッサおよび前記第2のプロセッサから参照可能なメモリ領域は、前記第2のプロセッサと同一のプロセッサ要素内のメモリに配置されることを特徴とする付記3に記載のロードモジュール生成方法。

#### 【0073】

(付記5) 複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成プログラムにおいて、  
第1のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第1のメモリ空間構築工程と、  
前記第1のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第1のプロセッサにより実行されるプログラム中の各シンボルのアド

レスを算出する第1のメモリ空間内アドレス解決工程と、

第2のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第2のメモリ空間構築工程と、

前記第2のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第2のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第2のメモリ空間内アドレス解決工程と、

前記第1のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第2のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決工程と、

をコンピュータに実行させることを特徴とするロードモジュール生成プログラム。

#### 【0074】

(付記6) 前記メモリ空間間アドレス解決工程では、前記第1のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第2のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスの、前記第2のメモリ空間構築工程で構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする付記5に記載のロードモジュール生成プログラム。

#### 【0075】

(付記7) 前記第1のメモリ空間構築工程では、前記第1のプロセッサからのみ参照可能なメモリ領域と、前記第1のプロセッサおよび前記第2のプロセッサから参照可能なメモリ領域とを構築することを特徴とする付記5または付記6に記載のロードモジュール生成プログラム。

#### 【0076】

(付記8) 前記第1のプロセッサおよび前記第2のプロセッサから参照可能なメモリ領域は、前記第2のプロセッサと同一のプロセッサ要素内のメモリに配置されることを特徴とする付記7に記載のロードモジュール生成プログラム。

#### 【0077】

(付記9) 複数のプロセッサを搭載する計算機システムにより実行されるプログ

ラムのロードモジュールを生成するロードモジュール生成装置において、

第1のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第1のメモリ空間構築手段と、

前記第1のメモリ空間構築手段により構築された各メモリ領域のアドレスにもとづいて、前記第1のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第1のメモリ空間内アドレス解決手段と、

第2のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第2のメモリ空間構築手段と、

前記第2のメモリ空間構築手段により構築された各メモリ領域のアドレスにもとづいて、前記第2のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第2のメモリ空間内アドレス解決手段と、

前記第1のメモリ空間内アドレス解決手段によりアドレスが解決されなかったシンボルのアドレスを、前記第2のメモリ空間内アドレス解決手段により算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決手段と、

を備えたことを特徴とするロードモジュール生成装置。

#### 【0078】

(付記10) 前記メモリ空間間アドレス解決手段は、前記第1のメモリ空間内アドレス解決手段によりアドレスが解決されなかったシンボルのアドレスを、前記第2のメモリ空間内アドレス解決手段により算出された当該シンボルのアドレスの、前記第2のメモリ空間構築手段により構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする付記9に記載のロードモジュール生成装置。

#### 【0079】

(付記11) 前記第1のメモリ空間構築手段は、前記第1のプロセッサからのみ参照可能なメモリ領域と、前記第1のプロセッサおよび前記第2のプロセッサから参照可能なメモリ領域とを構築することを特徴とする付記9または付記10に記載のロードモジュール生成装置。

#### 【0080】

(付記12) 前記第1のプロセッサおよび前記第2のプロセッサから参照可能なメモリ領域は、前記第2のプロセッサと同一のプロセッサ要素内のメモリに配置されることを特徴とする付記11に記載のロードモジュール生成装置。

#### 【0081】

##### 【発明の効果】

以上説明したように本発明によれば、MPMDプログラミングにもとづいて作成された各プロセッサ用のソースプログラムから、分散共有メモリ型マルチプロセッサ方式を採用する計算機でも実行可能なそのロードモジュールを生成することができるので（逆にいえば、分散共有メモリ型マルチプロセッサ方式の計算機についても、MPMDプログラミングによってプログラムを作成することができる）、これによって、分散共有メモリ型マルチプロセッサ方式を採用する計算機システムにおいても、より少ないメモリ容量で動作するプログラムのロードモジュールを生成することが可能なロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置が得られるという効果を奏する。

##### 【図面の簡単な説明】

###### 【図1】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムを模式的に示す説明図である。

###### 【図2】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムにおける、メモリ空間の定義例を模式的に示す説明図である。

###### 【図3】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、SPMDプログラミングにもとづくプログラムの一例を示す説明図である。

###### 【図4】

図3に示したプログラムのロードモジュールの生成手順を示すフローチャートである。

###### 【図5】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムを模式的に示す説明図である。

## 【図6】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムにおける、メモリ空間の定義例を模式的に示す説明図である。

## 【図7】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、MPMDプログラミングにもとづくプログラムの一例（PE #0用）を示す説明図である。

## 【図8】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、MPMDプログラミングにもとづくプログラムの一例（PE #1～#n用）を示す説明図である。

## 【図9】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、MPMDプログラミングにもとづくプログラムの一例（PE #0用）を示す説明図である。

## 【図10】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、MPMDプログラミングにもとづくプログラムの一例（PE #1用）を示す説明図である。

## 【図11】

図9および図10に示したプログラムが本発明によりアドレス解決された後の状況を模式的に示す説明図である。

## 【図12】

本発明の実施の形態にかかるロードモジュール生成装置のハードウェア構成の一例を示すブロック図である。

## 【図13】

本発明の実施の形態にかかるロードモジュール生成装置の構成を機能的に示す

ブロック図である。

【図14】

メモリ空間定義情報記憶部1306に保持されるメモリ空間定義情報の内容を模式的に示す説明図である。

【図15】

本発明の実施の形態にかかるロードモジュール生成装置における、図9および図10に示したプログラムのロードモジュール生成処理の手順を示すフローチャートである。

【図16】

本発明の実施の形態にかかるロードモジュール生成装置における、図9および図10に示したプログラムのロードモジュール生成処理の手順を示すフローチャートである。

【符号の説明】

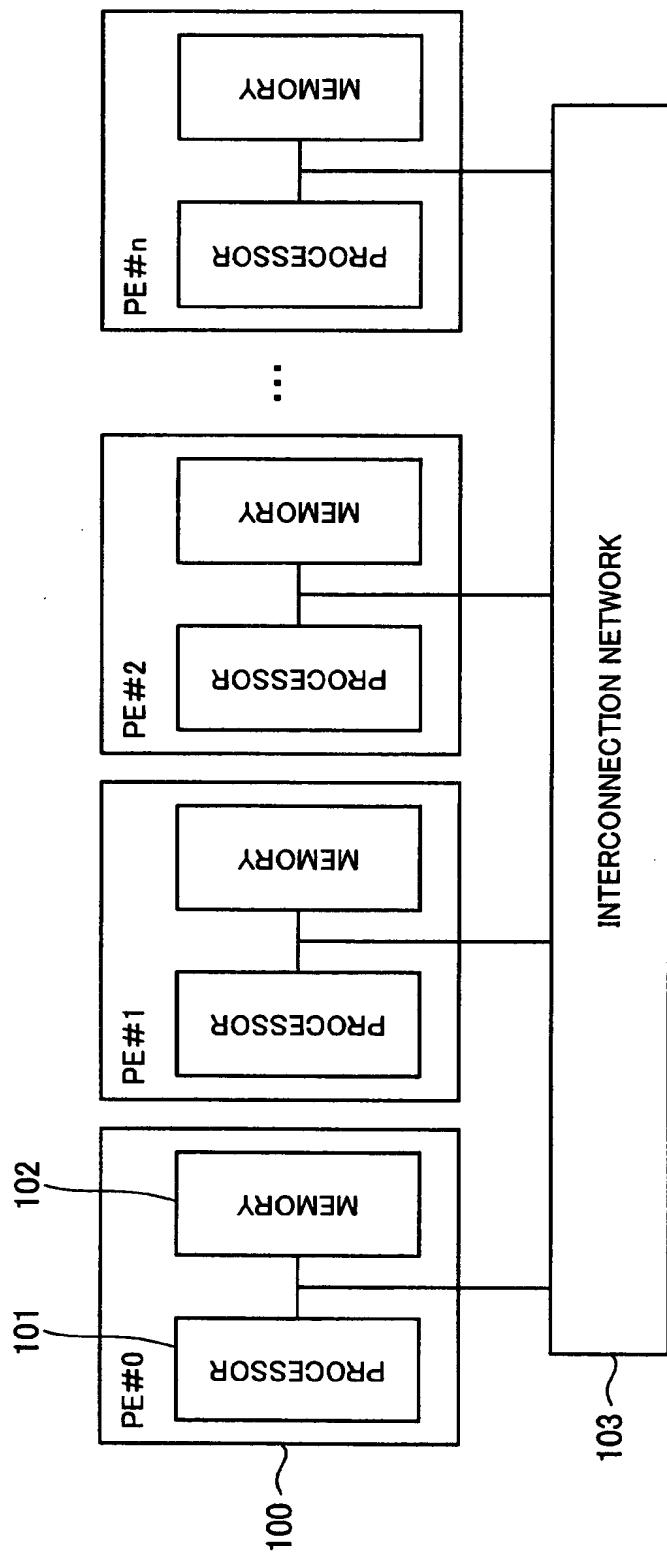
- 1200 バスまたはケーブル
- 1201 CPU
- 1202 ROM
- 1203 RAM
- 1204 HDD
- 1205 HD
- 1206 FDD
- 1207 FD
- 1208 ディスプレイ
- 1209 ネットワークI/F
- 1210 イーサネット(R)ケーブル
- 1211 キーボード
- 1212 マウス
- 1300 第1解析部
- 1301 命令列生成部
- 1302 アセンブリ記述出力部

- 1303 第2解析部
- 1304 バイナリ・コード生成部
- 1305 オブジェクト出力部
- 1306 メモリ空間定義情報記憶部
- 1307 オブジェクト読み込み部
- 1308 メモリ空間構築部
- 1309 メモリ空間内アドレス解決部
- 1310 メモリ空間間アドレス解決部
- 1311 ロードモジュール出力部

【書類名】図面

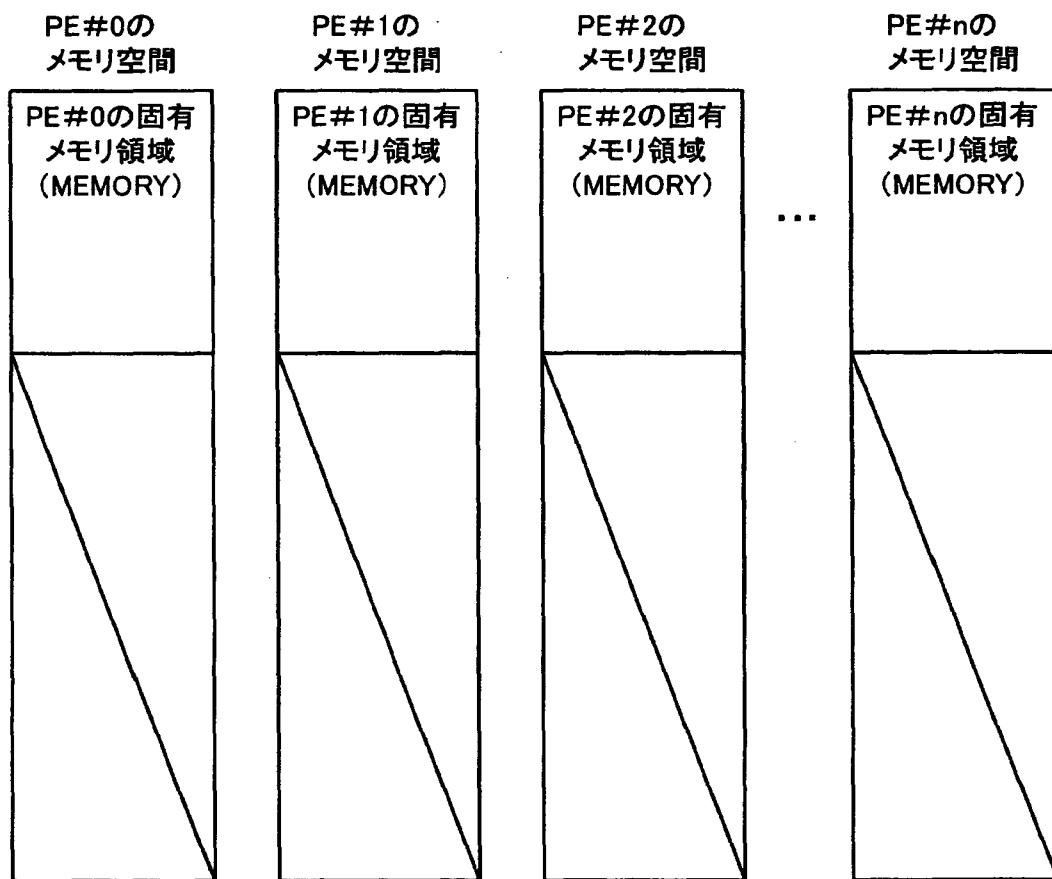
【図1】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムを模式的に示す説明図



【図2】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムにおける、  
メモリ空間の定義例を模式的に示す説明図



## 【図3】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、  
SPMDプログラミングにもとづくプログラムの一例を示す説明図

```
#include <stdio.h>
#include <string.h>
#include "mpi.h"

int
main(int argc, char **argv)
{
    int my_rank; /* カレントプロセスのランク */
    int source; /* 送信プロセスのランク */
    int dest; /* 受信プロセスのランク */
    int tag=0; /* メッセージのタグ */
    char message[100]; /* メッセージの保管場所 */
    MPI_Status status; /* 受信の戻りステータス */

    /* MPI のスタートアップ */
    MPI_Init(&argc, argv);

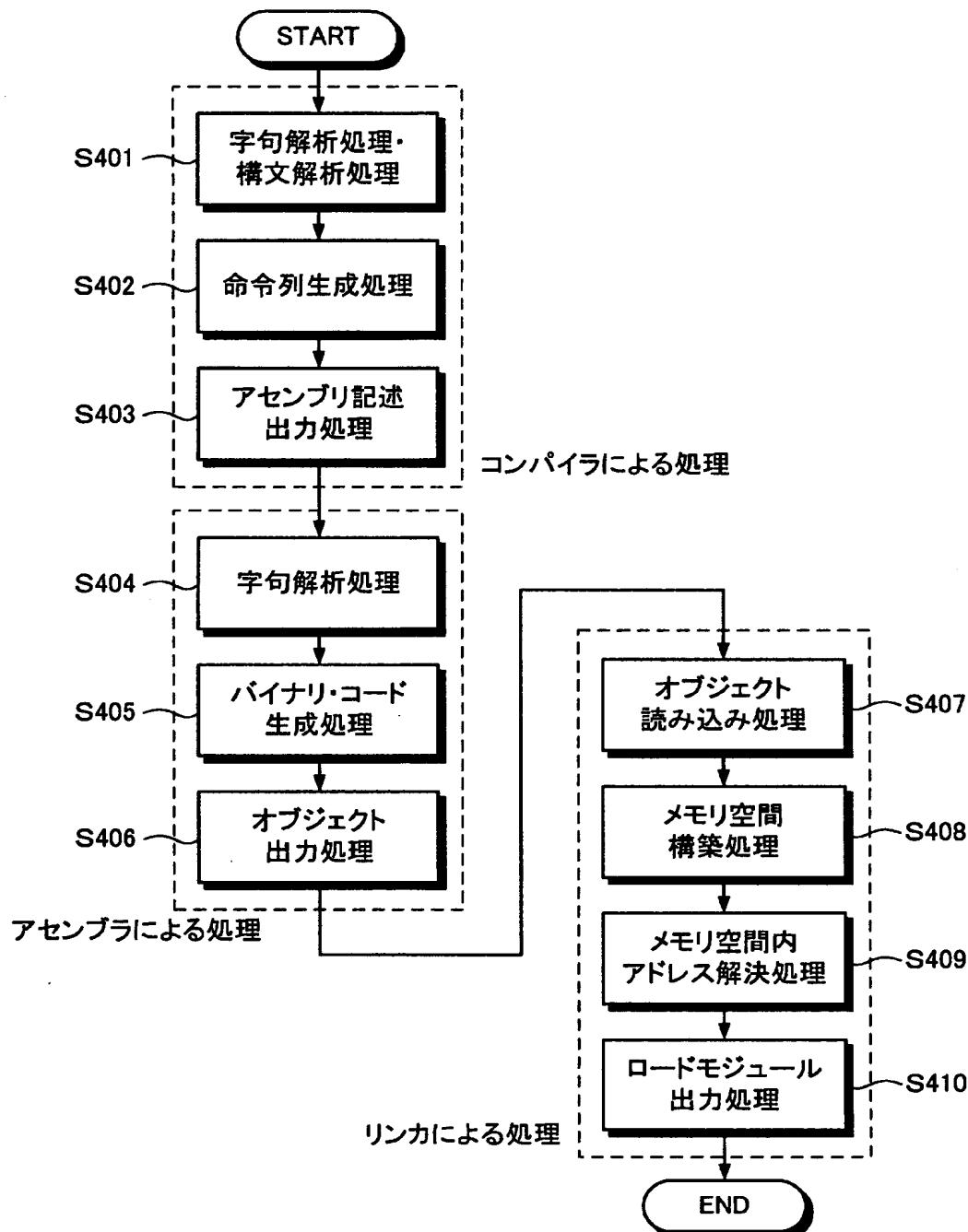
    /* カレントプロセスのランクを求める */
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    if (my_rank != 0) {
        /* メッセージの生成 */
        sprintf(message, "Greetings from process %d\n", my_rank);
        dest=0;
        /* '\0' も送信するので strlen+1 を使う */
        MPI_Send(message, strlen(message)+1, MPI_CHAR, dest, tag,
                 MPI_COMM_WORLD);
    } else {
        source=1;
        MPI_Recv(message, sizeof(message), MPI_CHAR, source, tag,
                 MPI_COMM_WORLD, &status);
        printf("%s\n", message);
    }

    /* MPI のシャットダウン */
    MPI_Finalize();
    return 0;
}
```

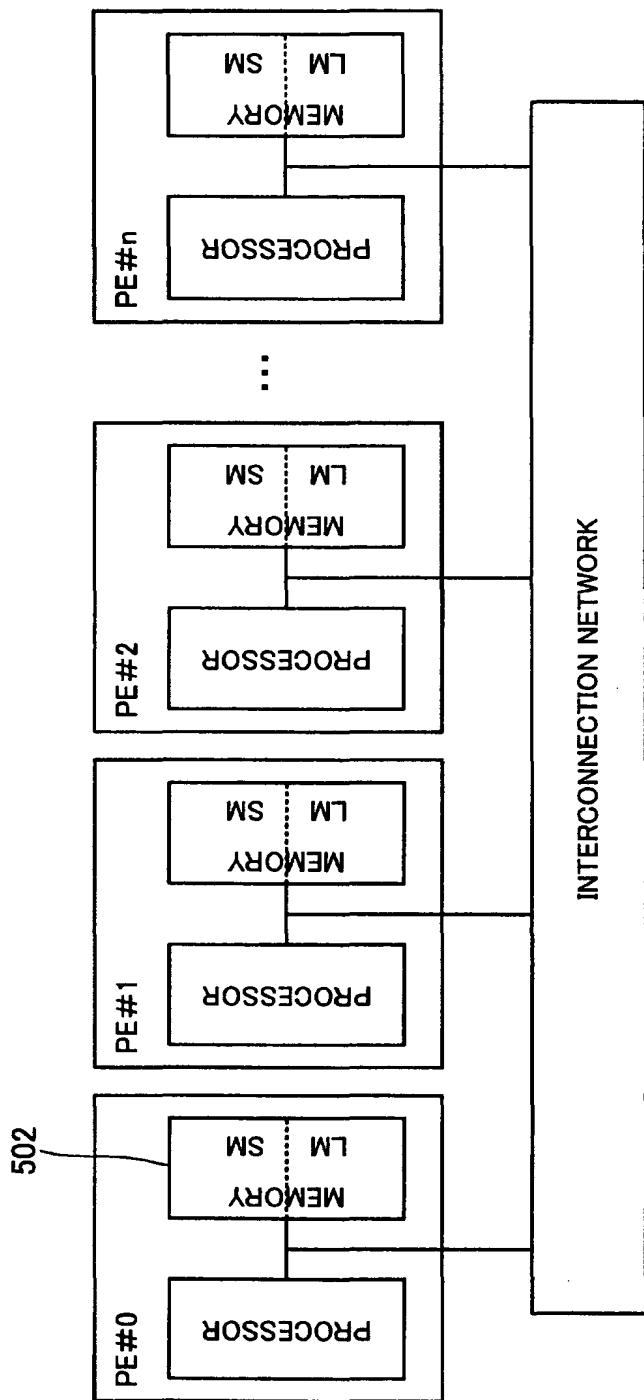
【図4】

図3に示したプログラムのロードモジュールの生成手順を示すフローチャート



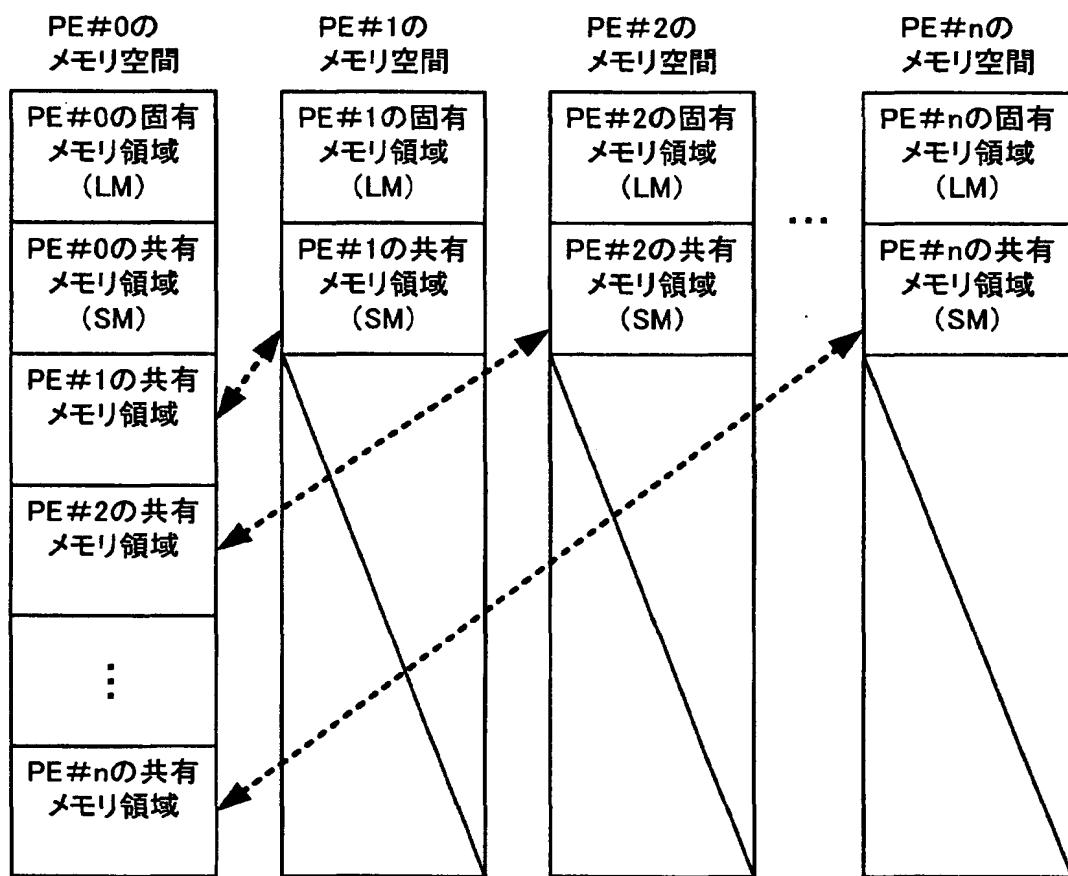
【図5】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムを模式的に示す説明図



【図6】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムにおける、  
メモリ空間の定義例を模式的に示す説明図



## 【図7】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、  
MPMDプログラミングにもとづくプログラムの一例(PE#0用)を示す説明図

```
#include <stdio.h>
#include <string.h>
#include "mpi.h"

int
main(int argc, char **argv)
{
    int my_rank; /* カレントプロセスのランク */
    int source; /* 送信プロセスのランク */
    int tag=0; /* メッセージのタグ */
    char message[100]; /* メッセージの保管場所 */
    MPI_Status status; /* 受信の戻りステータス */

    /* MPI のスタートアップ */
    MPI_Init(&argc, argv);

    /* カレントプロセスのランクを求める */
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    source=1;
    MPI_Recv(message, sizeof(message), MPI_CHAR, source, tag,
             MPI_COMM_WORLD, &status);
    printf("%s\n", message);

    /* MPI のシャットダウン */
    MPI_Finalize();
    return 0;
}
```

## 【図8】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、  
MPMDプログラミングにもとづくプログラムの一例(PE#1～#n用)を示す説明図

```
#include <stdio.h>
#include <string.h>
#include "mpi.h"

int
main(int argc, char **argv)
{
    int my_rank; /* カレントプロセスのランク */
    int dest; /* 受信プロセスのランク */
    int tag=0; /* メッセージのタグ */
    char message[100]; /* メッセージの保管場所 */

    /* MPI のスタートアップ */
    MPI_Init(&argc, argv);

    /* カレントプロセスのランクを求める */
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    /* メッセージの生成 */
    sprintf(message, "Greetings from process %d\n", my_rank);
    dest=0;
    /* '\0' も送信するので strlen+1 を使う */
    MPI_Send(message, strlen(message)+1, MPI_CHAR, dest, tag,
             MPI_COMM_WORLD);

    /* MPI のシャットダウン */
    MPI_Finalize();
    return 0;
}
```

## 【図9】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、  
MPMDプログラミングにもとづくプログラムの一例(PE#0用)を示す説明図

```
int input;
int output;
extern int in;
extern int out;

void
Th0(void)
{
    MOVE(&in, &input, sizeof(in));      /* Th0-1 */
    START(1.“Th1”);                  /* Th0-2 */
    MOVE(&output, &out, sizeof(output)); /* Th0-3 */
}
```

## 【図10】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、  
MPMDプログラミングにもとづくプログラムの一例(PE#1用)を示す説明図

```
int in;
int out;

void
Th1(void)
{
    extern void f1(int *, int *);

    f1(&in, &out);                  /* Th1-1 */
}
```

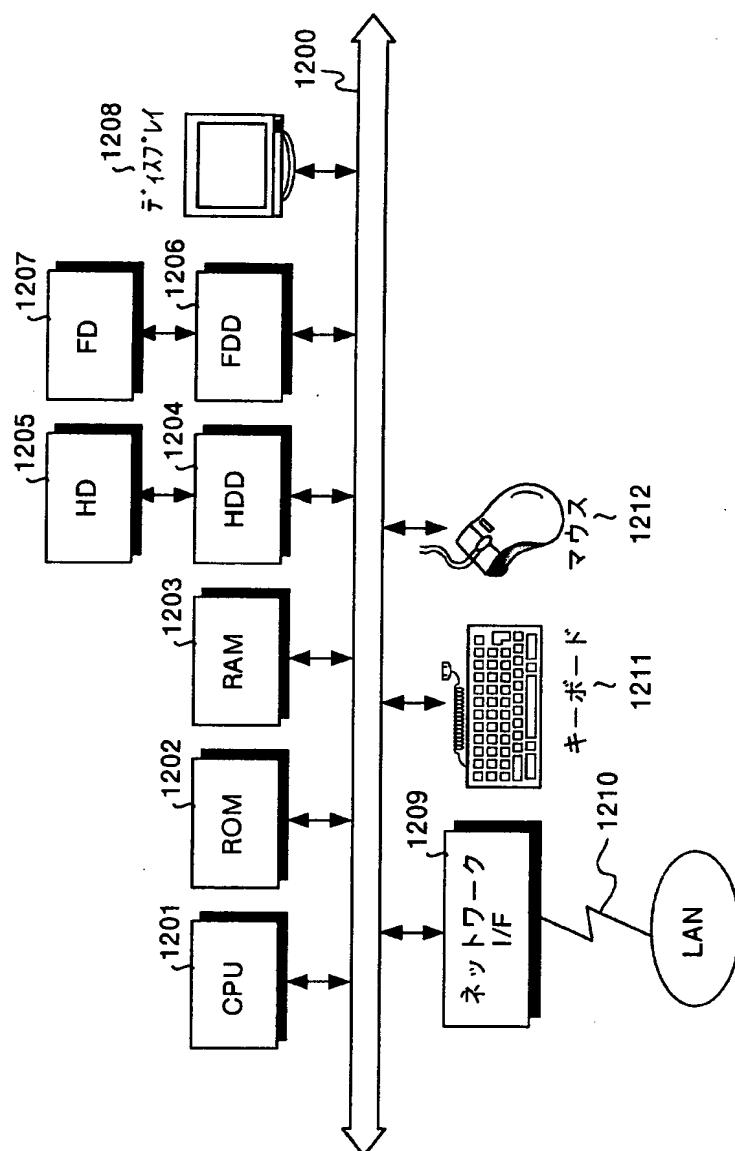
【図11】

図9および図10に示したプログラムが本発明によりアドレス解決された後の  
状況を模式的に示す説明図

PE#0のメモリ空間		PE#1のメモリ空間	
アドレス	内容	アドレス	内容
text area 0x0000	<pre>void Th0(void) {     MOVE(0x3000, 0x1000, sizeof(in));     START(1, "Th1");     MOVE(0x1004, 0x3004, sizeof(output)); }</pre>	0x0000	<pre>void Th1(void) {     f1(0x2000, 0x2004); }  void f1(int *in, int, *out) {     ... }</pre>
data area 0x1000	int input;	0x1000	
shared data area #0 0x2000			
shared data area #1 0x3000	int in;	0x2000	int in;
	0x3004	0x2004	int out;

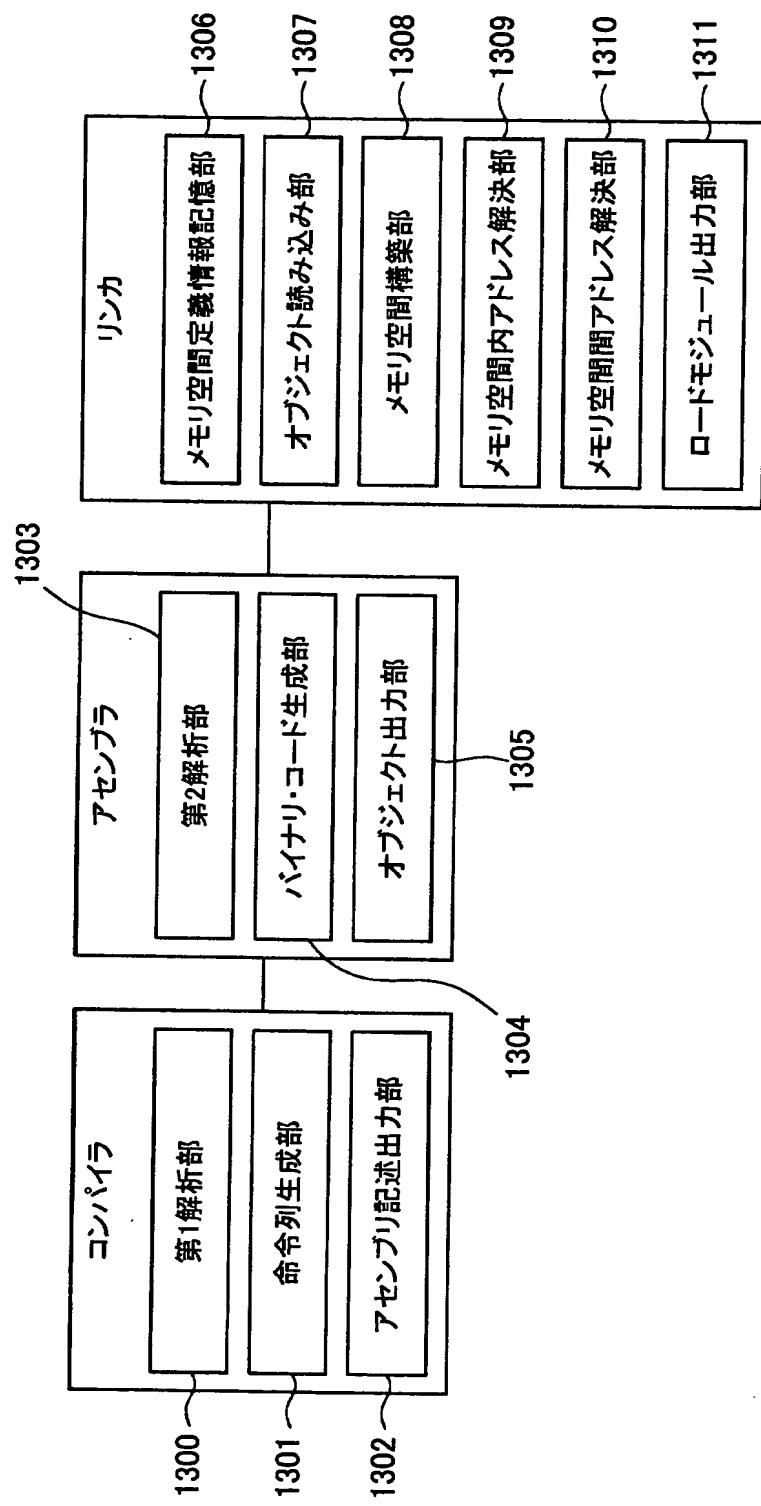
【図12】

本発明の実施の形態にかかるロードモジュール生成装置の  
ハードウェア構成の一例を示すブロック図



【図13】

本発明の実施の形態にかかるロードモジュール生成装置の構成を機能的に示すブロック図

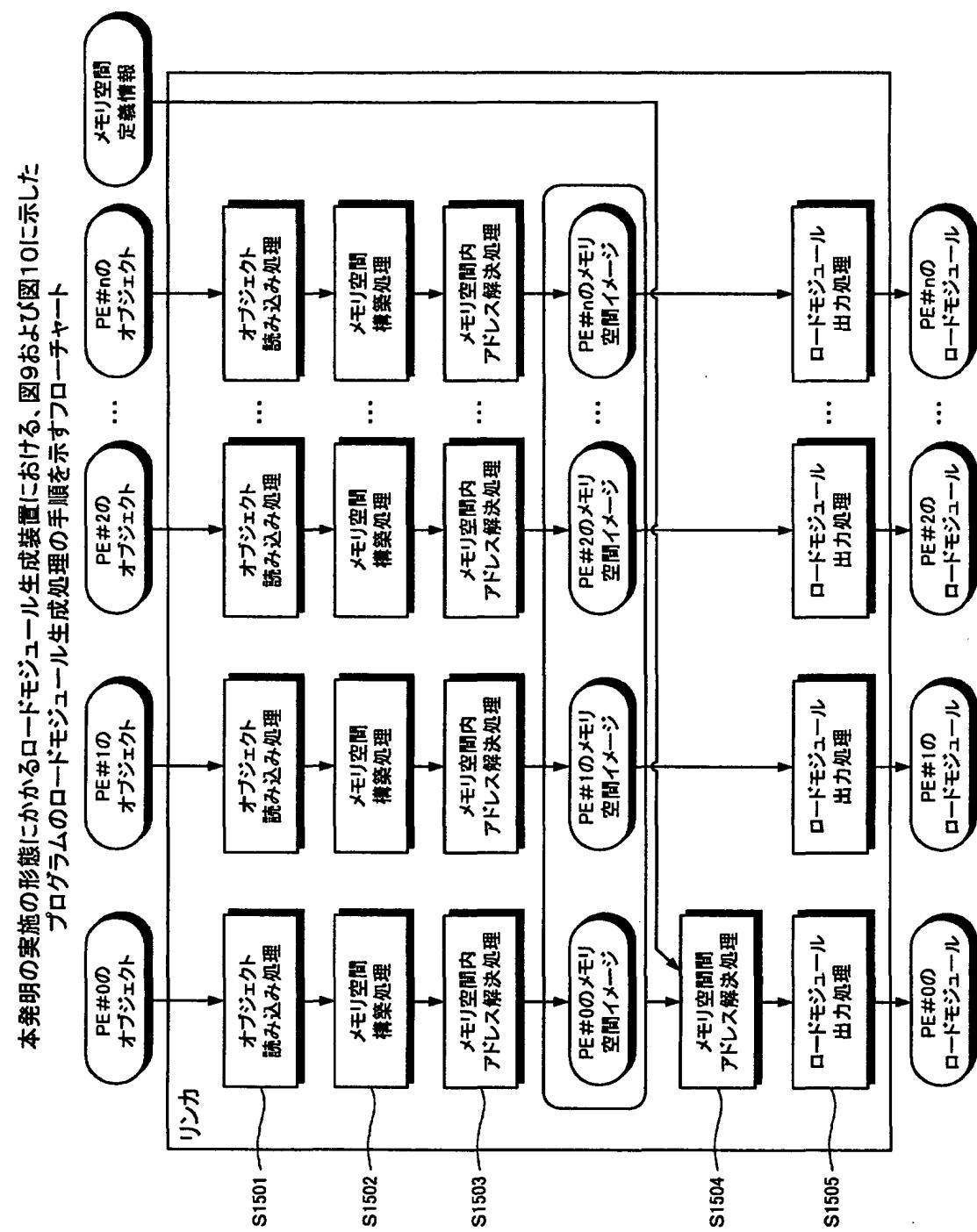


【図14】

メモリ空間定義情報記憶部1306に保持されるメモリ空間定義情報の内容を模式的に示す説明図

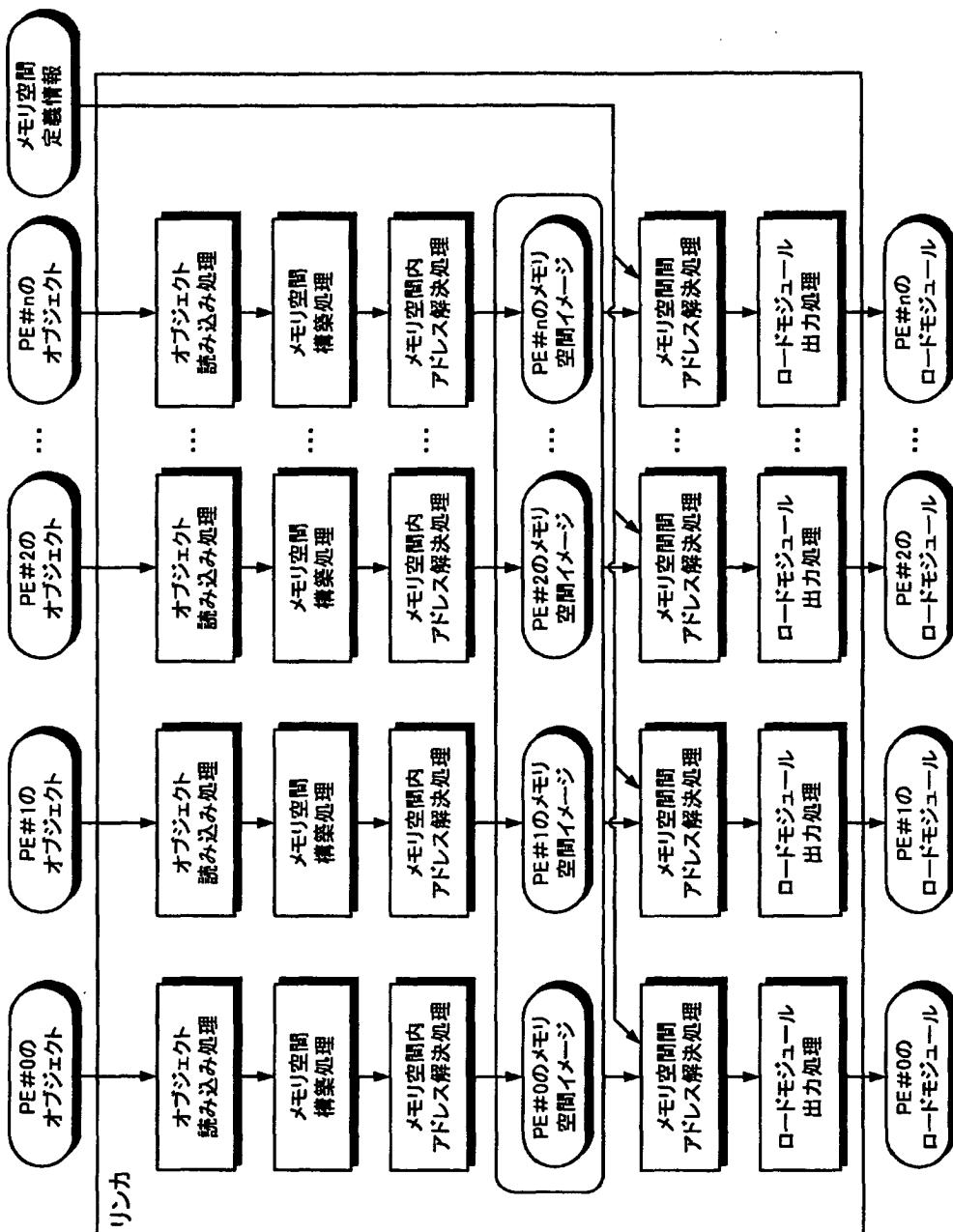
PE識別	領域名	開始アドレス	終了アドレス
PE#0	text area	0x0000	0xffff
	data area	0x1000	0x1fff
	shared data area #0	0x2000	0x2fff
	shared data area #1	0x3000	0x3fff
PE#1	text area	0x0000	0xffff
	data area	0x1000	0x1fff
	shared data area #1	0x2000	0x2fff

【図15】



【図16】

本発明の実施の形態にかかるロードモジュール生成装置における、図9および図10に示したプログラムのロードモジュール生成処理の手順を示すフローチャート



【書類名】 要約書

【要約】

【課題】 分散共有メモリ型マルチプロセッサ方式を採用する計算機システムにおいて、S P M D でなく M P M D プログラミングにもとづくプログラムを動作させることでメモリの有効利用をはかるべく、現実に実行可能な（すなわち、プログラム中の全シンボルのアドレスが解決された）当該プログラムのロードモジュールを生成するための言語処理系を提供すること。

【解決手段】 分散共有メモリ型マルチプロセッサ方式を前提とする M P M D プログラミングでは、たとえば P E # 0 用のプログラムが P E # 1 の共有メモリ上のデータを参照・変更することがあるが、当該データのアドレス（メモリ空間上の位置）は P E ごとに異なっている。そこでメモリ空間間アドレス解決部 1310 により、P E # 1 にとっての上記データのアドレスから、その P E # 0 にとってのアドレスを所定の計算式にもとづいて算出する。

【選択図】 図 1 3

出願人履歴情報

識別番号 [000005223]

1. 変更年月日 1996年 3月26日

[変更理由] 住所変更

住 所 神奈川県川崎市中原区上小田中4丁目1番1号  
氏 名 富士通株式会社